Tema 2.3: Red de Hebb

Clasificación de patrones con redes neuronales simples

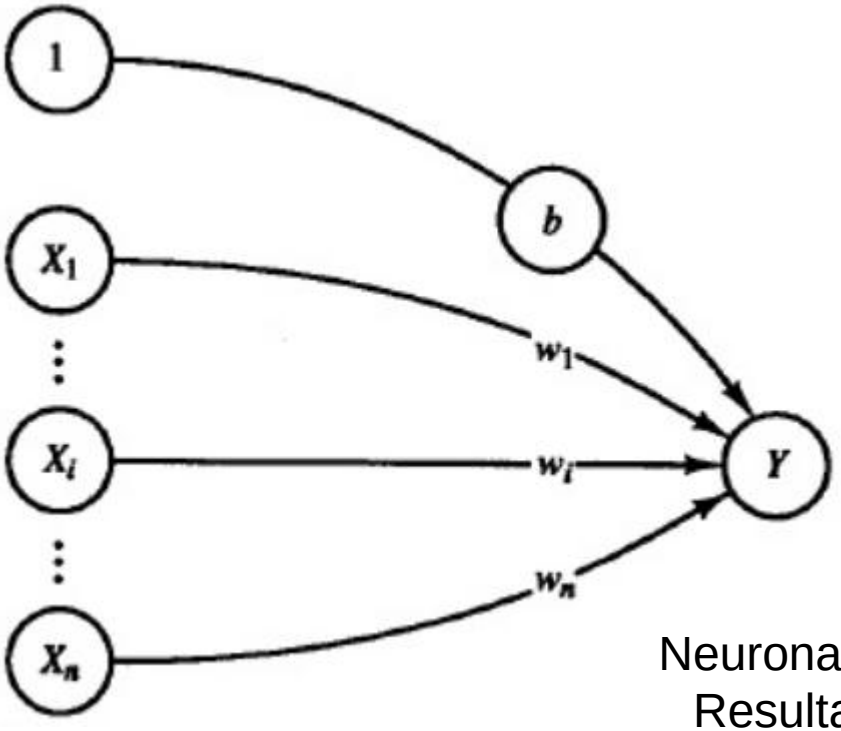
• En los problemas de clasificación de patrones, cada vector de entrada (patrón) pertenece o no a una clase o categoría.

• Se parte de un conjunto de patrones para los cuales se conoce su clasificación.

• La neurona de salida representa la pertenencia a una clase (por ejemplo con respuesta 1). La respuesta 0 ó -1 indica que el patrón no es miembro de la clase.

• La clasificación de patrones es un tipo de reconocimiento de patrones

Redes Neuronales de una sola capa para la clasificación de patrones



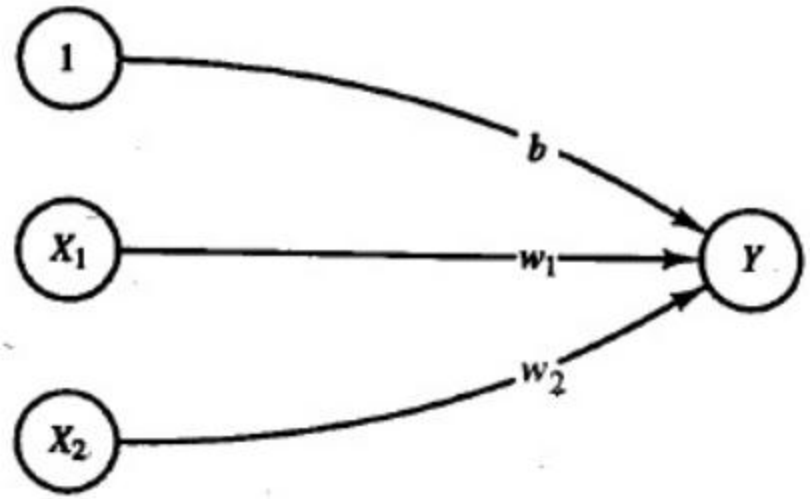
$$y\_in = b + \sum_i x_i w_i$$

$$f(y\_in) = \begin{cases} 1 & \text{si } y\_in \geq \theta \\ -1 & \text{si } y\_in < \theta \end{cases}$$

Capa de entrada: parámetros que se utilizan para la clasificación

Neurona de salida: Resultado de la clasificación

Sesgos b y umbrales θ



Capa de entrada: parámetros que se utilizan para la clasificación

Neurona de salida: Resultado de la clasificación

Opción 1

$$y\_in = b + \sum_i x_i w_i$$

$$f(y\_in) = \begin{cases} 1 & \text{si } y\_in \geq 0 \\ -1 & \text{si } y\_in < 0 \end{cases}$$

Opción 2

$$y\_in = \sum_i x_i w_i$$

$$f(y\_in) = \begin{cases} 1 & \text{si } y\_in \geq \theta \\ -1 & \text{si } y\_in < \theta \end{cases}$$

Vamos a demostrar que l a opción 2 es equivalente a la opción 1 en este tipo de red

Opción 1: con sesgo y $\theta = 0$

$$y\_in = b + \sum_i x_i w_i$$

$$f(y\_in) = \begin{cases} 1 & \text{si } y\_in \geq 0 \\ -1 & \text{si } y\_in < 0 \end{cases}$$

• Queremos separar el espacio de entradas en regiones donde la respuesta de la red es positiva y en regiones donde la respuesta de la red es negativa.

• La frontera entre valores de x_1 y x_2 para los cuales hay una respuesta positiva y valores para los cuales hay una respuesta negativa es la línea:

$$b + x_1 w_1 + x_2 w_2 = 0$$

• Si $w_2 \neq 0$ se puede expresar como: $$x_2 = -\frac{w_1}{w_2} x_1 - \frac{b}{w_2}$$

Opción 2: sin sesgo y $\theta \neq 0$

$$y\_in = \sum_i x_i w_i$$

$$f(y\_in) = \begin{cases} 1 & \text{si } y\_in \geq \theta \\ -1 & \text{si } y\_in < \theta \end{cases}$$

• Queremos separar el espacio de entradas en regiones donde la respuesta de la red es positiva y en regiones donde la respuesta de la red es negativa.

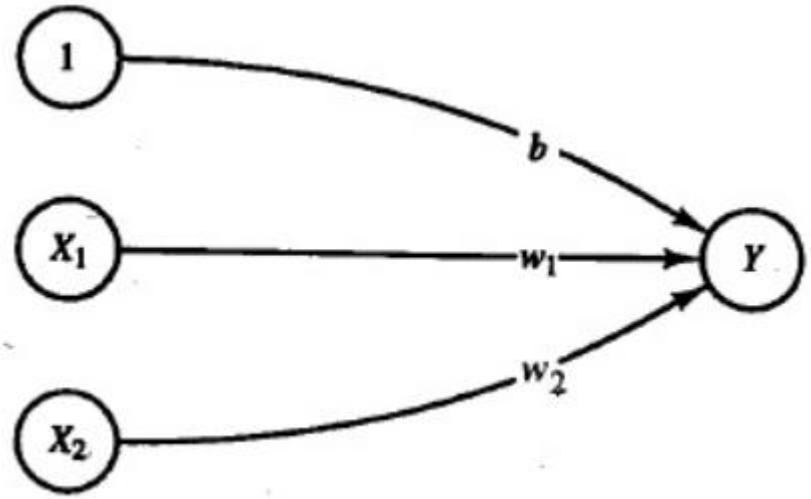
• La frontera entre valores de x_1 y x_2 para los cuales hay una respuesta positiva y valores para los cuales hay una respuesta negativa es la línea:

$$x_1 w_1 + x_2 w_2 = \theta$$

• Si $w_2 \neq 0$ se puede expresar como: $$x_2 = -\frac{w_1}{w_2} x_1 - \frac{\theta}{w_2}$$

La opción 2 es equivalente a la opción 1 en este tipo de red

Redes Neuronales de una sola capa para la clasificación de patrones



$$y\_in = b + \sum_i x_i w_i$$

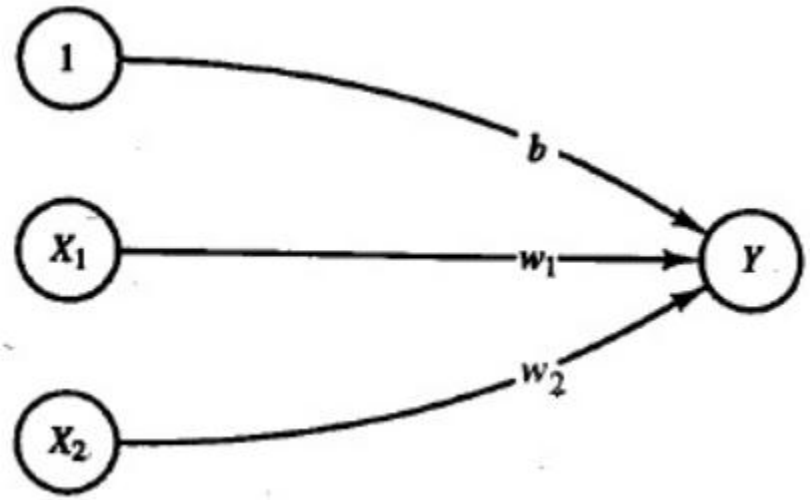
$$f(y\_in) = \begin{cases} 1 & \text{si } y\_in \geq 0 \\ -1 & \text{si } y\_in < 0 \end{cases}$$

• Durante el aprendizaje de la red se determinan los valores de w_1 y w_2 y b para que haya una respuesta correcta con los datos del entrenamiento.

• Si el sesgo (o el umbral) no es cero, entonces la línea de decisión no pasa por el origen (pero se pueden encontrar líneas que pasen tan cerca como se requiera).

• Si no se incluye ni el sesgo ni el umbral la línea de separación de la decisión pasa por el origen (esto limita mucho el tipo de clasificación que se puede hacer).

Separabilidad lineal (I)



$$y\_in = b + \sum_i x_i w_i$$

$$f(y\_in) = \begin{cases} 1 & \text{si } y\_in \geq 0 \\ -1 & \text{si } y\_in < 0 \end{cases}$$

• El objetivo es entrenar la red (determinar los pesos) para que realice la clasificación deseada cuando se presenta un patrón de entrada con el que se ha entrenado o un patrón que se parece a alguno de los patrones que se han utilizado para entrenar.

• En este ejemplo, si se reconoce un patrón la neurona de salida tiene un *output* de 1 y de -1 en caso contrario (codificación bipolar).

• La frontera de decisión la determina: $b + \sum_i x_i w_i = 0$

que puede ser una línea recta, un plano o un hiperplano dependiendo del número de neuronas de entrada de la red

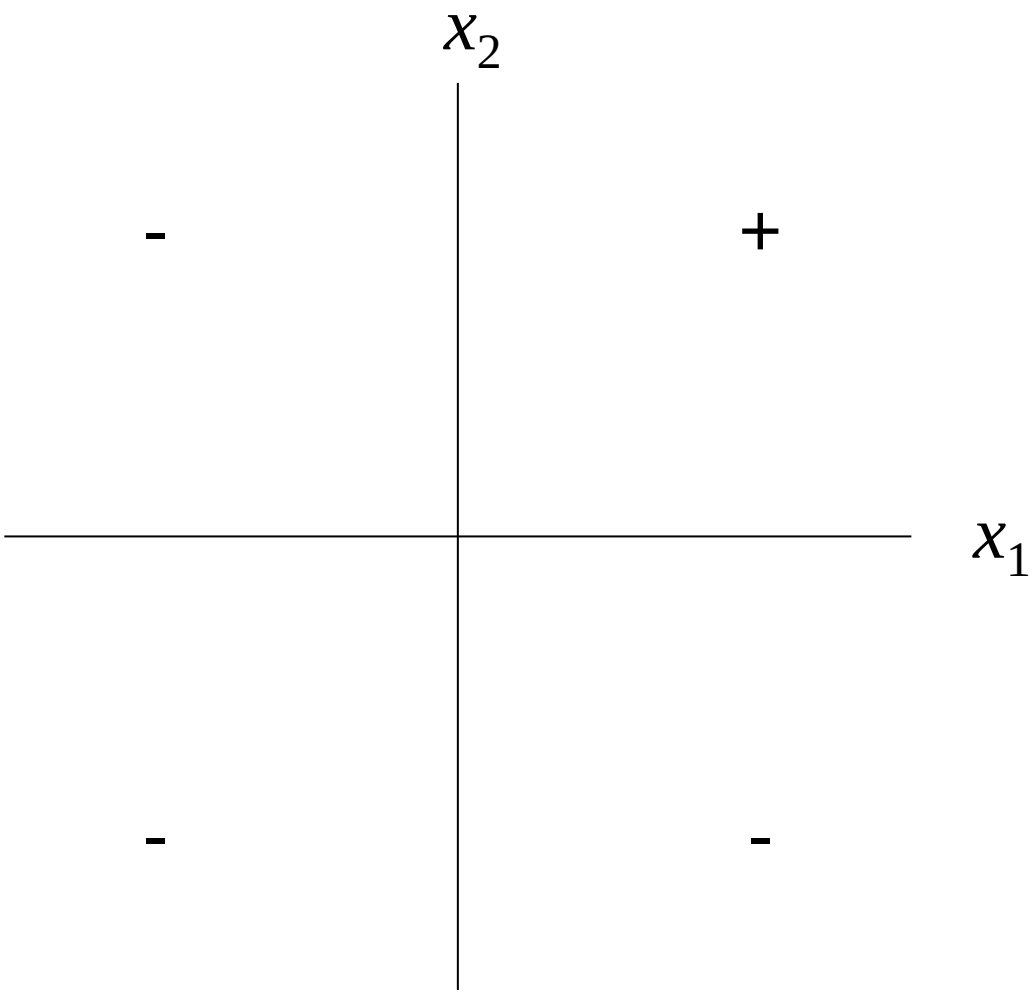
Separabilidad lineal (II)

- Se dice que el problema a resolver por la red es **separable linealmente** si existen pesos y sesgos de forma que todos los vectores de entrada del entrenamiento para los cuales la respuesta de la red es +1 están a un lado de la frontera de decisión y todos los vectores de entrada del entrenamiento para los cuales la respuesta es -1 están al otro lado de esta frontera.

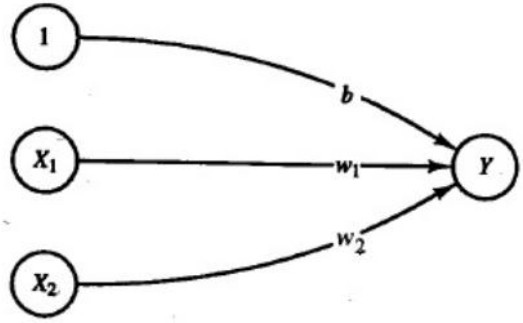
- Minksy y Papert demostraron en 1988 que una red de una sola capa sólo puede resolver problemas que son separables linealmente. De este resultado se deriva también que las redes multicapa que tienen activaciones lineales no resuelven problemas que no puedan resolver las redes mono-capa (la combinación de funciones lineales es una función lineal).

Respuesta deseada para la función AND

(inputs bipolares)

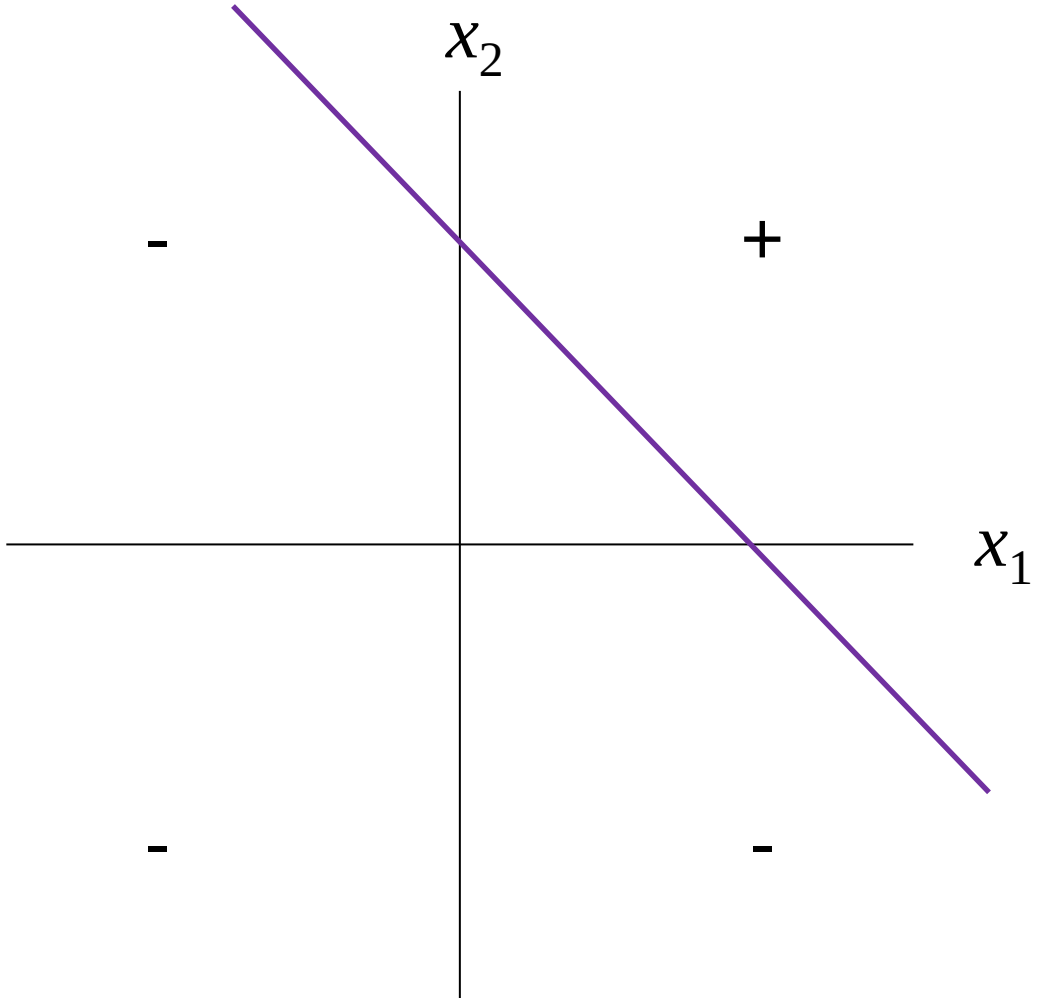


x_1	x_2	y
1	1	1
1	-1	-1
-1	1	-1
-1	-1	-1



Frontera de decisión para la función AND

(inputs bipolares)



$$y\_in = b + \sum_i x_i w_i$$

$$f(y\_in) = \begin{cases} 1 & \text{si } y\_in \geq 0 \\ -1 & \text{si } y\_in < 0 \end{cases}$$

x_1	x_2	y
1	1	1
1	-1	-1
-1	1	-1
-1	-1	-1

Ejemplo de posibles pesos

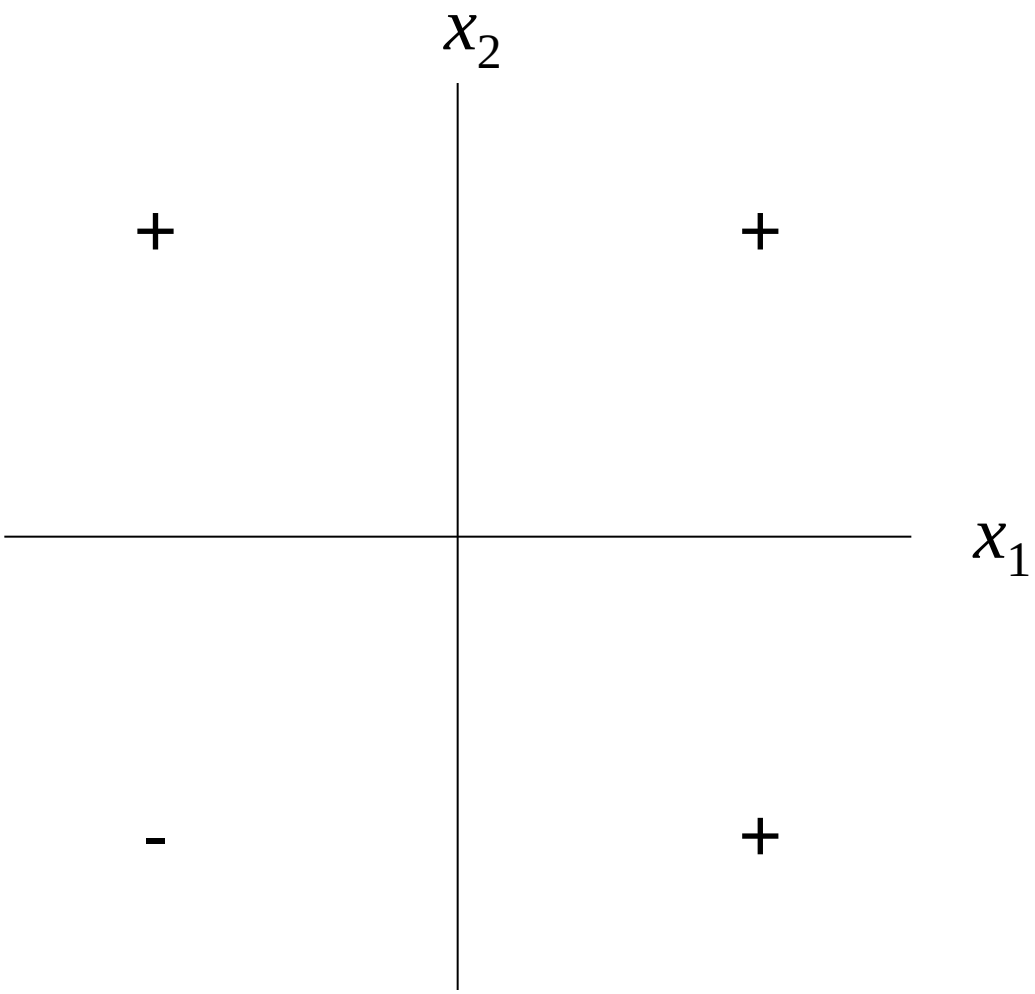
$b = -1$; $w_1=1$; $w_2=1$

La frontera de decisión puede ser la recta:

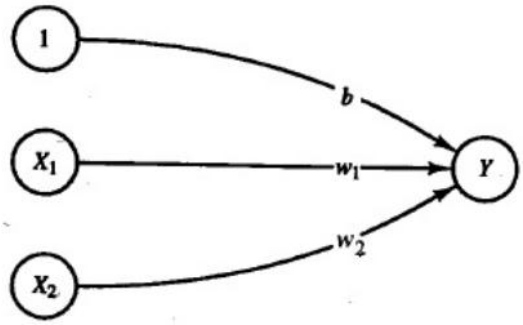
$x_2 = -x_1 + 1$

Respuesta deseada para la función OR

(inputs bipolares)

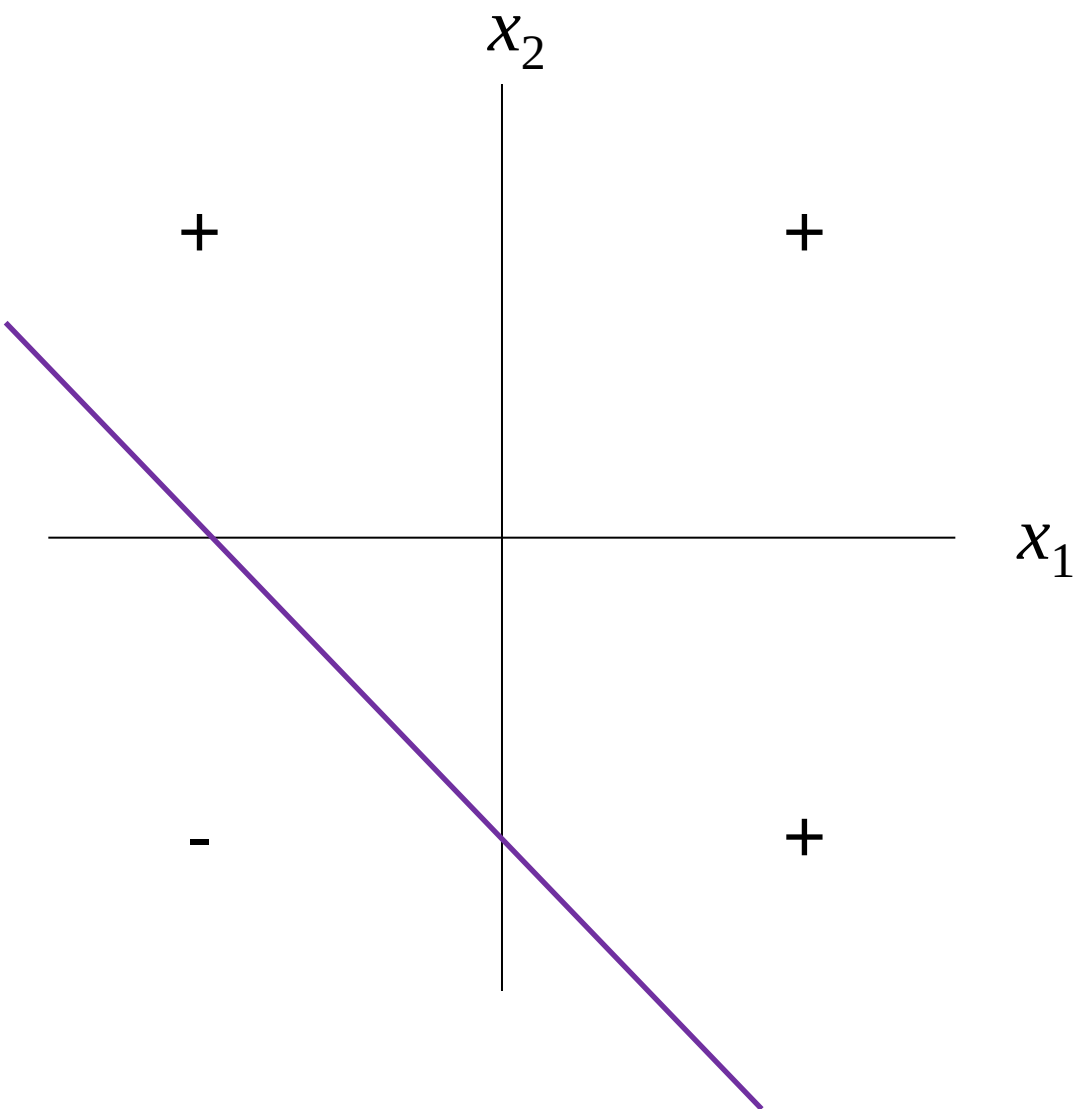


x_1	x_2	y
1	1	1
1	-1	1
-1	1	1
-1	-1	-1



Frontera de decisión para la función OR

(inputs bipolares)



x_1	x_2	y
1	1	1
1	-1	1
-1	1	1
-1	-1	-1

Ejemplo de posibles pesos

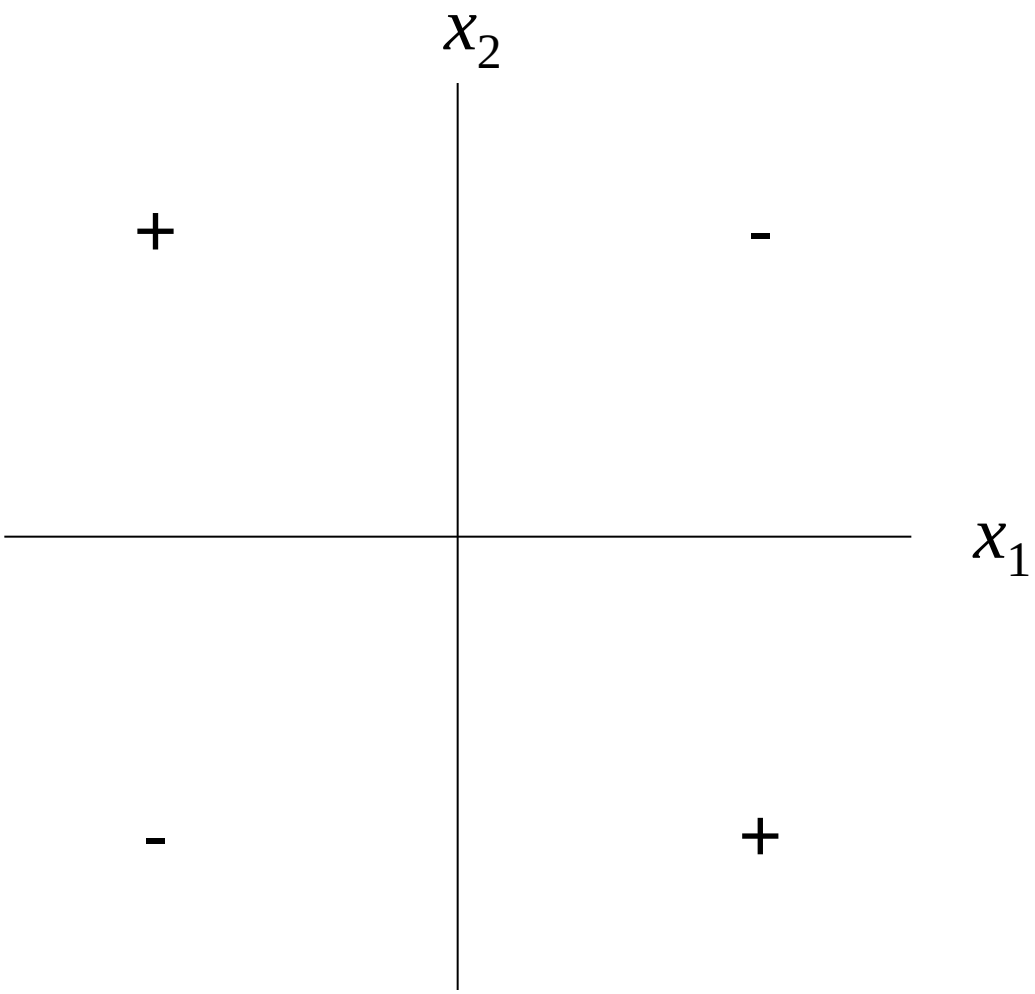
$b = 1; w_1=1; w_2=1$

La frontera de decisión puede ser la recta:

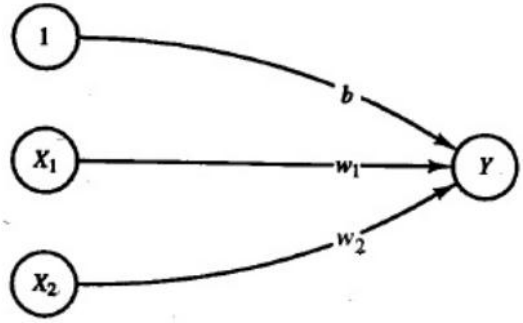
$x_2 = -x_1 - 1$

Respuesta deseada para la función XOR

(inputs bipolares)



x_1	x_2	y
1	1	-1
1	-1	1
-1	1	1
-1	-1	-1



Representación binaria o bipolar de los datos de entrenamiento

- La representación de los datos puede determinar si se puede resolver o no un problema con una red neuronal específica.

- En general, con la representación binaria de los datos la red puede generalizar peor, puesto que en la representación bipolar los datos desconocidos se pueden representar como 0 y por tanto se pueden distinguir de datos incorrectos (1 cambiado por -1 o al revés).

Red de Hebb: red de una sola capa con regla de aprendizaje de Hebb

- Recordamos la regla de aprendizaje de Hebb: “Si dos neuronas están conectadas sinápticamente y sus disparos están relacionados causalmente, se refuerza la conexión”

- Aquí la vamos a extender para que también se refuerce la conexión si las dos neuronas no disparan al mismo tiempo.

$$w_i(\text{nuevo}) = w_i(\text{anterior}) + x_i y$$

Algoritmo de aprendizaje de la red de Hebb

Paso 0: Inicializar todos los pesos $w_i=0 \quad (i=1,..,n)$

Paso 1: Para cada par de entrenamiento vector de entrada y salida objetivo (**s**:t), ejecutar los pasos 2-4:

Paso 2: Establecer las activaciones de las neuronas de entrada:

$$x_i = s_i \quad (i=1 \ldots n)$$

Paso 3: Estableces las activaciones de la neurona de salida:

$$y = t$$

Paso 4: Ajustar los pesos y el sesgo según la ley de Hebb:

$$w_i(\text{nuevo}) = w_i(\text{anterior}) + x_i y \qquad (i=1 \ldots n)$$

$$b\ (\text{nuevo}) = b(\text{anterior}) + y$$

La actualización de los pesos se puede indicar también con la siguiente notación:

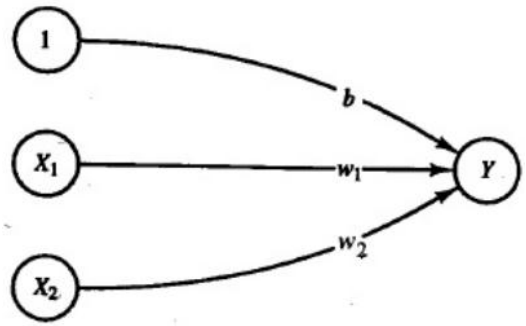
$$\mathbf{w}\ (\text{nuevo}) = \mathbf{w}\ (\text{anterior}) + \mathbf{x}y$$

y se suele denotar $\Delta\mathbf{w} = \mathbf{x}y$, por lo que la actualización queda:

$$\mathbf{w}\ (\text{nuevo}) = \mathbf{w}\ (\text{anterior}) + \Delta\mathbf{w}$$

Ejemplos de uso de la red de Hebb con su regla de aprendizaje

Función AND con red de Hebb
(entradas y objetivos binarios)



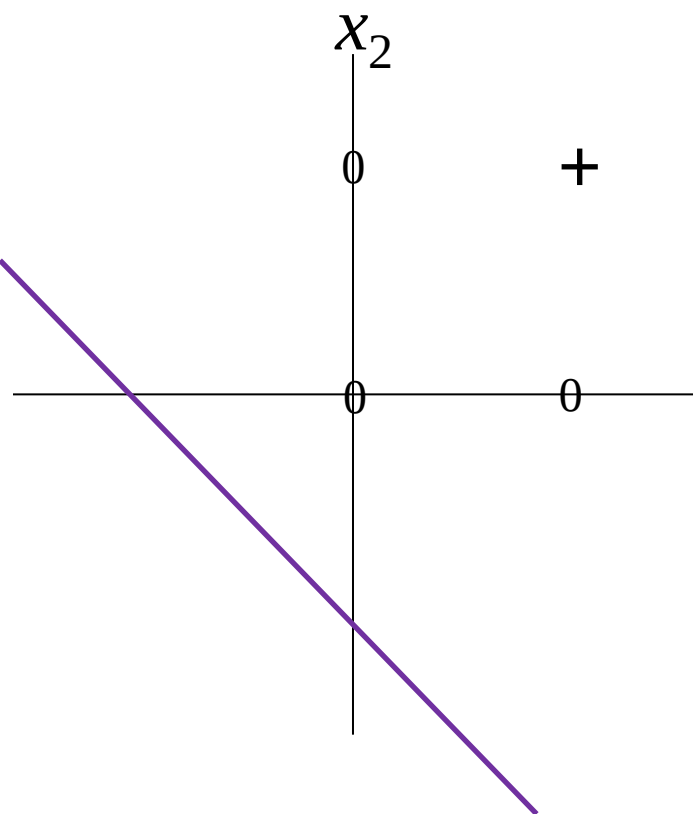
$$y\_in = b + \sum_i x_i w_i$$

$$f(y\_in) = \begin{cases} 1 & \text{si } y\_in \geq 0 \\ 0 & \text{si } y\_in < 0 \end{cases}$$

w_i(nuevo) = w_i(anterior) + $x_i y$

b (nuevo) = b(anterior) + y

			t				0	0	0
x_1	x_2	b	y	Δw_1	Δw_2	Δb	w_1	w_2	b
1	1	1	1	1	1	1	1	1	1
1	0	1	0	0	0	0	1	1	1
0	1	1	0	0	0	0	1	1	1
0	0	1	0	0	0	0	1	1	1



$b = 1$; w_1=1; w_2=1

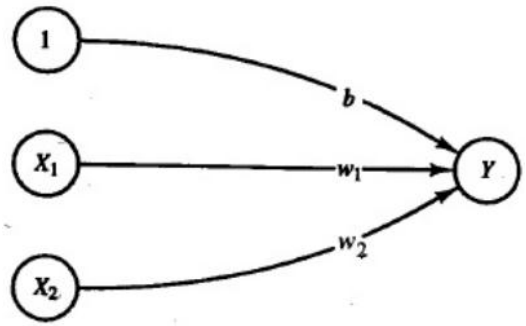
Obtenemos la recta:

x_2 =-x_1 -1

$$x_2 = -\frac{w_1}{w_2} x_1 - \frac{b}{w_2}$$

No hay aprendizaje a partir del segundo par de entrenamiento. El problema son los ceros, la codificación no parece ser buena. Vamos a probar otra.

Función AND con red de Hebb
(entradas binarias y objetivos bipolares)



$$y\_in = b + \sum_i x_i w_i$$

$$f(y\_in) = \begin{cases} 1 & \text{si } y\_in \geq 0 \\ -1 & \text{si } y\_in < 0 \end{cases}$$

w_i(nuevo) = w_i(anterior) + $x_i y$

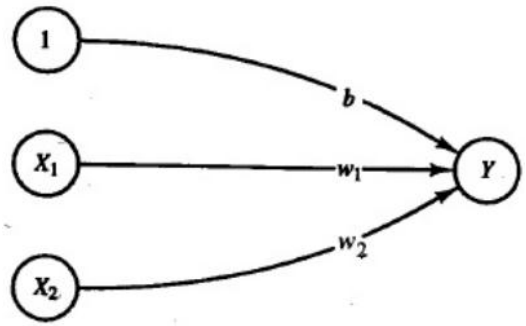
b (nuevo) = b(anterior) + y

			t				0	0	0
x_1	x_2	b	y	Δw_1	Δw_2	Δb	w_1	w_2	b
1	1	1	1	1	1	1	1	1	1
1	0	1	-1	-1	0	-1	0	1	0
0	1	1	-1	0	-1	-1	0	0	-1
0	0	1	-1	0	0	-1	0	0	-2

Con esta codificación tampoco se soluciona el problema.

Vamos a probar otra más.

Función AND con red de Hebb
(entradas y objetivos bipolares)



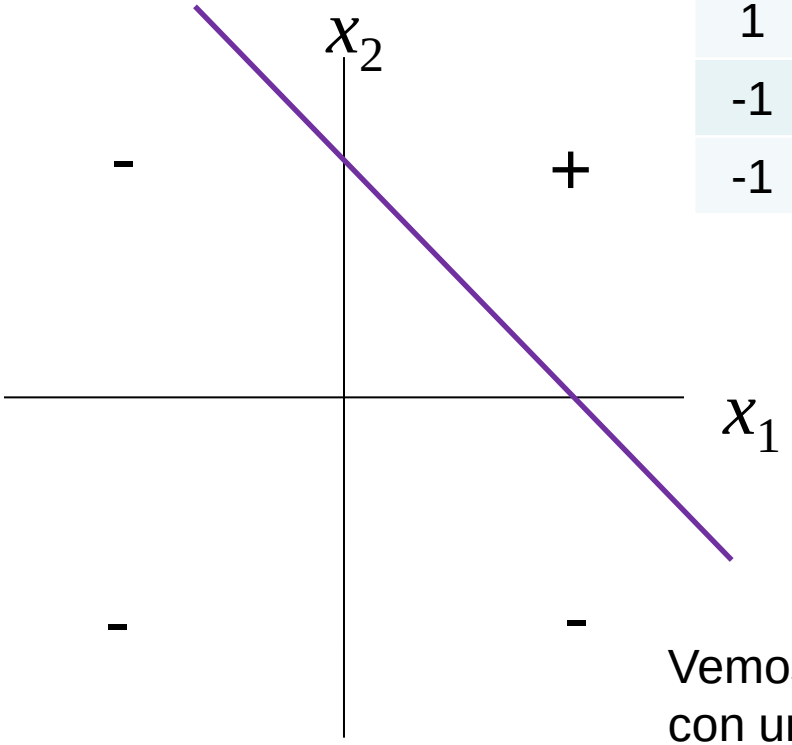
$$y\_in = b + \sum_i x_i w_i$$

$$f(y\_in) = \begin{cases} 1 & \text{si } y\_in \geq 0 \\ -1 & \text{si } y\_in < 0 \end{cases}$$

w_i(nuevo) = w_i(anterior) + $x_i y$

b (nuevo) = b(anterior) + y

			t				0	0	0
x_1	x_2	b	y	Δw_1	Δw_2	Δb	w_1	w_2	b
1	1	1	1	1	1	1	1	1	1
1	-1	1	-1	-1	1	-1	0	2	0
-1	1	1	-1	1	-1	-1	1	1	-1
-1	-1	1	-1	1	1	-1	2	2	-2



$b = -2$; $w_1 = 2$; $w_2 = 2$

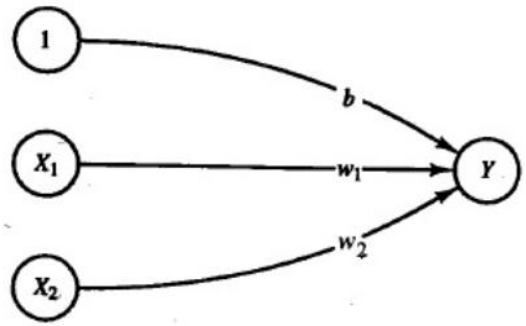
Obtenemos la recta:

$x_2 = -x_1 + 1$

$$\left(x_2 = -\frac{w_1}{w_2} x_1 - \frac{b}{w_2} \right)$$

Vemos que la regla de Hebb conduce a la solución pero con una codificación específica.

Función OR con red de Hebb
(entradas y objetivos bipolares)



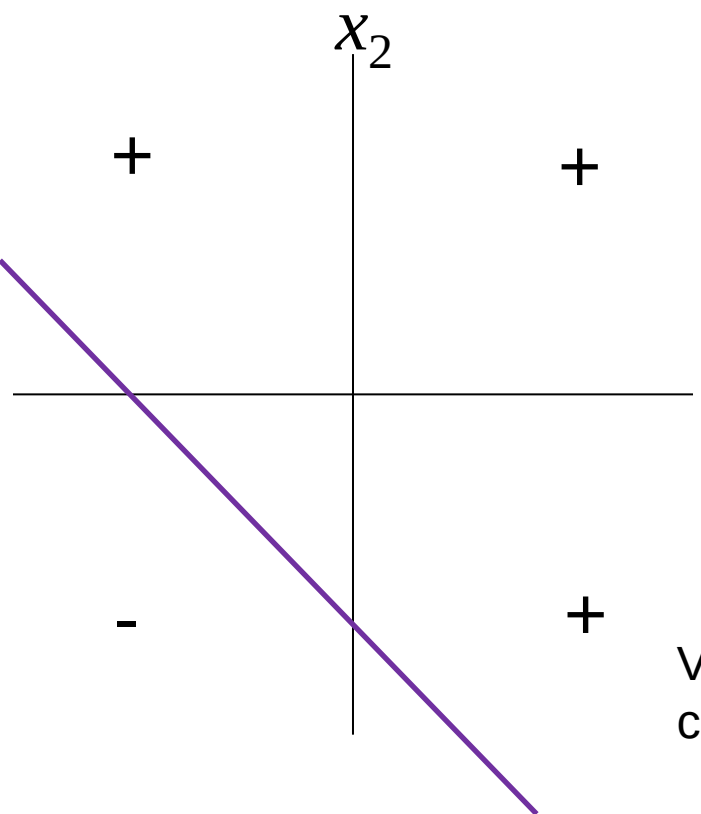
w_i(nuevo) = w_i(anterior) + $x_i y$

b (nuevo) = b(anterior) + y

$$y\_in = b + \sum_i x_i w_i$$

$$f(y\_in) = \begin{cases} 1 & \text{si } y\_in \geq 0 \\ -1 & \text{si } y\_in < 0 \end{cases}$$

			t				0	0	0
x_1	x_2	b	y	Δw_1	Δw_2	Δb	w_1	w_2	b
1	1	1	1	1	1	1	1	1	1
1	-1	1	1	1	-1	1	2	0	2
-1	1	1	1	-1	1	1	1	1	3
-1	-1	1	-1	1	1	-1	2	2	2



$b = 2$; w_1=2; w_2=2

Obtenemos la recta:

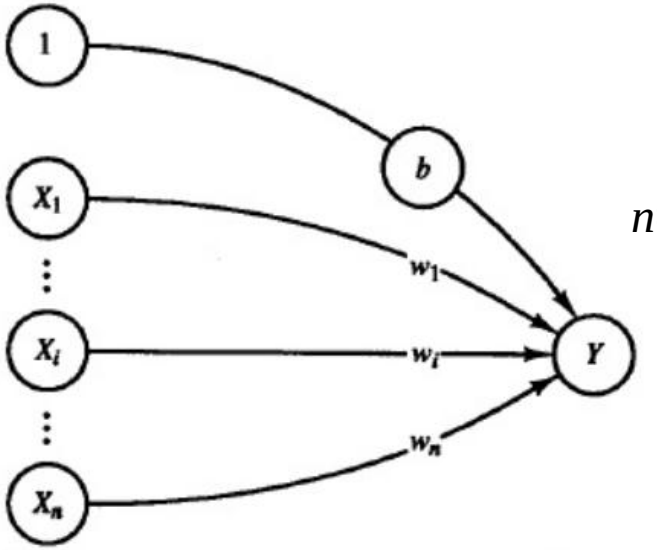
x_2 =-x_1 -1

$$\left(x_2 = -\frac{w_1}{w_2} x_1 - \frac{b}{w_2} \right)$$

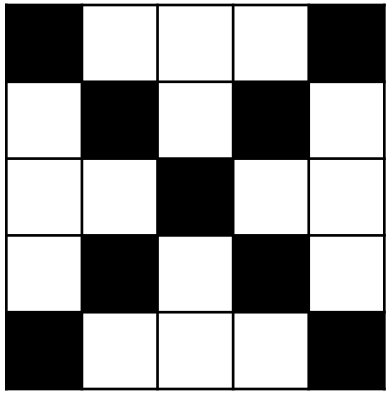
Vemos que la regla de Hebb conduce a la solución pero con una codificación específica.

Reconocimiento de caracteres con red de Hebb

Codificamos letras con pixeles en una rejilla de tamaño 5x5.
Utilizaremos codificación bipolar para las entradas y los objetivos

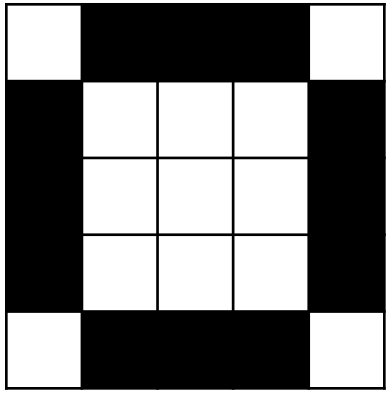


n=25



Entrada (1 -1 -1 -1 1, -1 1 -1 1 -1, -1 -1 1 -1 -1, -1 1 -1 1 -1, 1 -1 -1 -1 1)

Salida 1



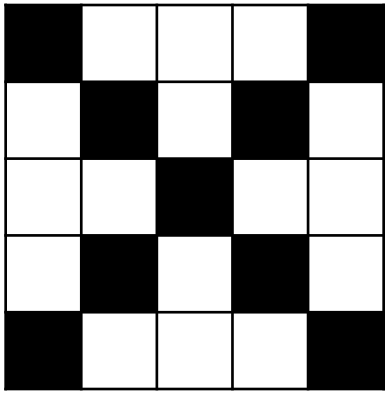
Entrada (-1 1 1 1 -1, 1 -1 -1 -1 1, 1 -1 -1 -1 1, 1 -1 -1 -1 1, -1 1 1 1 -1)

Salida -1

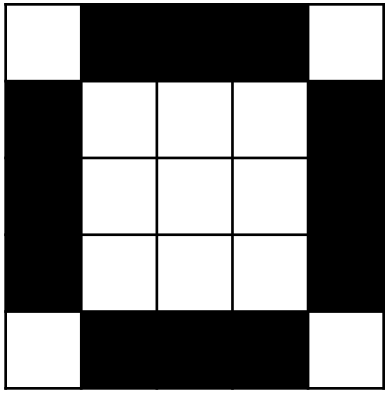
Reconocimiento de caracteres con red de Hebb

Para calcular los pesos multiplicamos cada una de las entradas por la activación de salida:

$$\mathbf{w}\,(\text{nuevo}) = \mathbf{w}\,(\text{anterior}) + \mathbf{x}y$$



Para la letra “x“ obtenemos los mismos valores de entrada (multiplicamos por 1 y sumamos los pesos iniciales -0-):
(1 -1 -1 -1 1, -1 1 -1 1 -1, -1 -1 1 -1 -1, -1 1 -1 1 -1, 1 -1 -1 -1 1) y para el sesgo 1



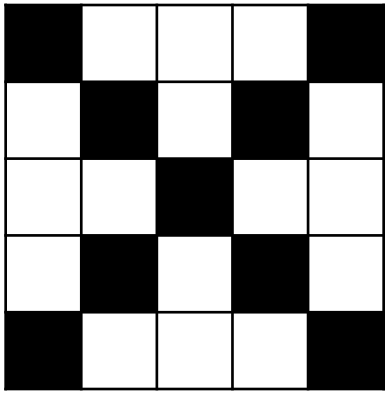
Para la letra “o” obtenemos los valores de entrada cambiados de signo (multiplicamos por -1)

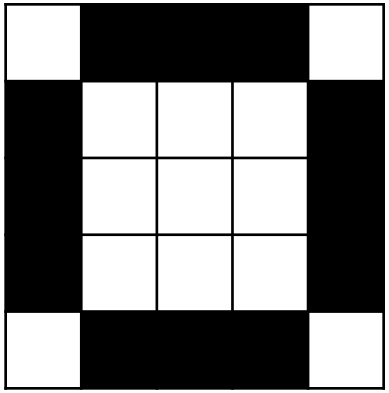
(1 -1 -1 -1 1, -1 1 1 1 -1, -1 1 1 1 -1, -1 1 1 1 -1, 1 -1 -1 -1 1) y para el sesgo -1

Sumandos los pesos anteriores, después del entrenamiento los pesos quedan:

(2 -2 -2 -2 2, -2 2 0 2 -2, -2 0 2 0 -2, -2 2 0 2 -2, 2 -2 -2 -2 2) y para el sesgo 0

Reconocimiento de caracteres con red de Hebb





Salida para los vectores de entrenamiento

Ahora podemos calcular la salida para cada uno de los patrones de entrenamiento. La entrada total es el producto escalar del patrón de entrada por el vector de pesos:

La entrada neta para el primer vector de entrenamiento sale 42 y para el segundo vector -42.

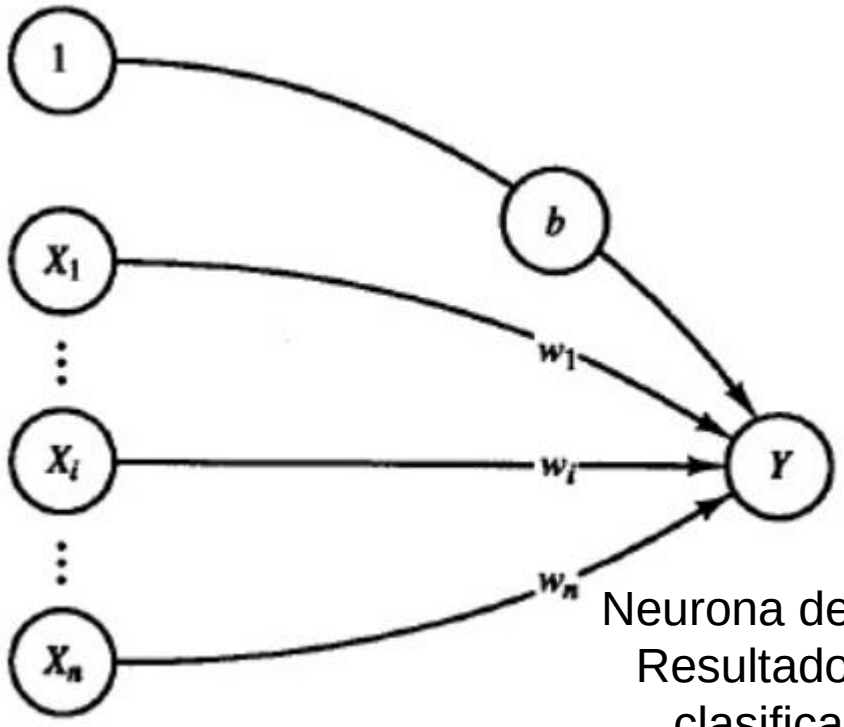
$$f(y\_in) = \begin{cases} 1 & \text{si } y\_in \geq 0 \\ -1 & \text{si } y\_in < 0 \end{cases}$$

El resultado para el primer vector de entrenamiento sale 1 y para el segundo vector -1.

Vectores de entrada parecidos también dan este mismo resultado.

Tema 2.4: Perceptrón

Perceptrón



Capa de entrada: parámetros que se utilizan para la clasificación

Neurona de salida: Resultado de la clasificación

$$y\_in = b + \sum_i x_i w_i$$

$$f(y\_in) = \begin{cases} 1 & \text{si } y\_in > \theta \\ 0 & \text{si } -\theta \leq y\_in \leq \theta \\ -1 & \text{si } y\_in < -\theta \end{cases}$$

$$w_i(\text{nuevo}) = w_i(\text{anterior}) + \alpha t x_i$$

t es el valor objetivo y α es la tasa de aprendizaje

Perceptrón

• Durante el aprendizaje, los patrones se pasan varias veces de forma iterativa

• La función de transferencia utiliza el umbral ($\theta \neq 0$) de forma distinta a la red de Hebb.

• La regla de aprendizaje incluye un factor o tasa de aprendizaje $\alpha \leq 1$

• Con hipótesis específicas, se puede probar teóricamente que los pesos convergen durante el entrenamiento.

Algoritmo de aprendizaje del perceptrón

Paso 0: Inicializar todos los pesos y sesgos (por simplicidad a cero)
Establecer la tasa de aprendizaje $\alpha\ (0 < \alpha \leq 1)$

Paso 1: Mientras que la condición de parada sea falsa, ejecutar pasos 2-6

Paso 2: Para cada par de entrenamiento ($\boldsymbol{s}$:t), ejecutar los pasos 3-5:

Paso 3: Establecer las activaciones a las neuronas de entrada

$$x_i = s_i \quad (i=1 \ldots n)$$

Paso 4: Calcular la respuesta de la neurona de salida:

$$y\_in = b + \sum_i x_i w_i$$

$$f(y\_in) = \begin{cases} 1 & \text{si } y\_in > \theta \\ 0 & \text{si } -\theta \leq y\_in \leq \theta \\ -1 & \text{si } y\_in < -\theta \end{cases}$$

Paso 5: Ajustar los pesos y el sesgo si ha ocurrido un error para este patrón:

Si $y \neq t$

$$w_i(\text{nuevo}) = w_i(\text{anterior}) + \alpha t x_i$$

$$b(\text{nuevo}) = b(\text{anterior}) + \alpha t$$

Si no

$$w_i(\text{nuevo}) = w_i(\text{anterior})$$

$$b(\text{nuevo}) = b(\text{anterior})$$

Paso 6: Comprobar la condición de parada: si no han cambiado los pesos en el paso 2: parar; en caso contrario, continuar.

Regla de aprendizaje del perceptrón

Si $y \neq t$

$$w_i(\text{nuevo}) = w_i(\text{anterior}) + \alpha t x_i$$
$$b(\text{nuevo}) = b(\text{anterior}) + \alpha t$$

- Sólo se actualizan los pesos que conectan neuronas de entrada activas ($x_i \neq 0$).

- Los pesos sólo se actualizan para patrones que no produzcan una salida correcta para y.

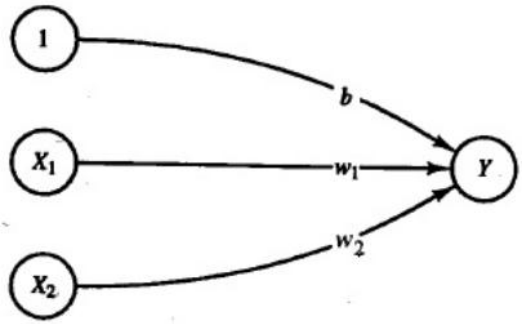
- Por tanto, a medida que más patrones de entrenamiento producen más salidas correctas, menos cambios de pesos hay.

- La función de transferencia implica una banda de indecisión.

- El uso de umbral y sesgo no es equivalente en el perceptrón.

Ejemplos de uso del perceptrón

Perceptrón para la función AND con entradas binarias y objetivos bipolares



$$y\_in = b + \sum_i x_i w_i$$

$$w_i(\text{nuevo}) = w_i(\text{anterior}) + \alpha t x_i$$
$$b\ (\text{nuevo}) = b(\text{anterior}) + \alpha t$$

$$f(y\_in) = \begin{cases} 1 & \text{si } y\_in > \theta \\ 0 & \text{si } -\theta \le y\_in \le \theta \\ -1 & \text{si } y\_in < -\theta \end{cases}$$

Pesos iniciales y sesgo = *0*
θ= 0.2, α=1
$\Delta w_i = \alpha t x_i$
$\Delta b = \alpha t$

1ª época

x_1	x_2	b	y_in	y	t	Δw_1	Δw_2	Δb	w_1	w_2	b
1	1	1	0	0	1	1	1	1	1	1	1
1	0	1	2	1	-1	-1	0	-1	0	1	0
0	1	1	1	1	-1	0	-1	-1	0	0	-1
0	0	1	-1	-1	-1	0	0	0	0	0	-1

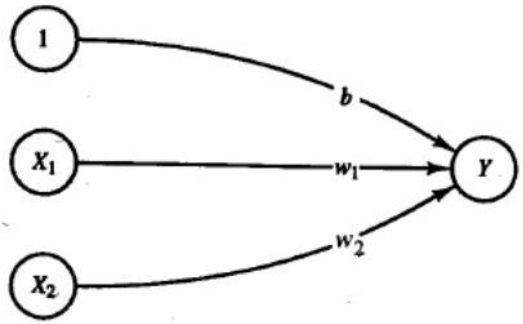
Los pesos han variado en la primera época, luego continuamos con otra.

2ª época

x_1	x_2	b	y_in	y	t	Δw_1	Δw_2	Δb	w_1	w_2	b
1	1	1	-1	-1	1	1	1	1	1	1	0
1	0	1	1	1	-1	-1	0	-1	0	1	-1
0	1	1	0	0	-1	0	-1	-1	0	0	-2
0	0	1	-2	-1	-1	0	0	0	0	0	-2

También varían en la 2ª

Perceptrón para la función AND con entradas binarias y objetivos bipolares



$w_i(\text{nuevo}) = w_i(\text{anterior}) + \alpha t x_i$

$b\ (\text{nuevo}) = b(\text{anterior}) + \alpha t$

$$y\_in = b + \sum_i x_i w_i$$

$$f(y\_in) = \begin{cases} 1 & \text{si } y\_in > \theta \\ 0 & \text{si } -\theta \le y\_in \le \theta \\ -1 & \text{si } y\_in < -\theta \end{cases}$$

Pesos de la 2ª época:
w_1=0, w_2=0, b=-2
θ= 0.2, α=1
$\Delta w_i = \alpha t x_i$
$\Delta b = \alpha t$

3ª época

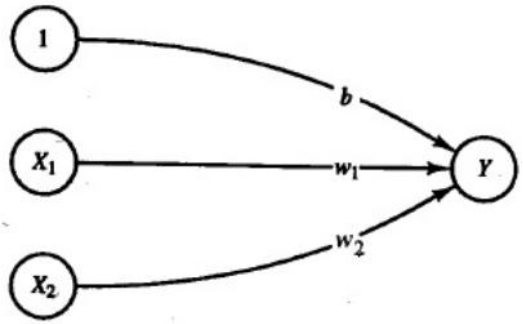
x_1	x_2	b	y_in	y	t	Δw_1	Δw_2	Δb	w_1	w_2	b
1	1	1	-2	-1	1	1	1	1	1	1	-1
1	0	1	0	0	-1	-1	0	-1	0	1	-2
0	1	1	-1	-1	-1	0	0	0	0	1	-2
0	0	1	-2	-1	-1	0	0	0	0	1	-2

Los pesos varían en la 3ª época y también en la 4ª

4ª época

x_1	x_2	b	y_in	y	t	Δw_1	Δw_2	Δb	w_1	w_2	b
1	1	1	-1	-1	1	1	1	1	1	2	-1
1	0	1	0	0	-1	-1	0	-1	0	2	-2
0	1	1	0	0	-1	0	-1	-1	0	1	-3
0	0	1	-3	-1	-1	0	0	0	0	1	-3

Perceptrón para la función AND con entradas binarias y objetivos bipolares



$$y\_in = b + \sum_i x_i w_i$$

$$f(y\_in) = \begin{cases} 1 & \text{si } y\_in > \theta \\ 0 & \text{si } -\theta \le y\_in \le \theta \\ -1 & \text{si } y\_in < -\theta \end{cases}$$

w_i(nuevo) = w_i(anterior) + $\alpha t x_i$

b (nuevo) = b(anterior) + αt

Pesos de la 3ª época:

w_1=0, w_2=1, b=-2

θ= 0.2, α=1

$\Delta w_i = \alpha t x_i$

$\Delta b = \alpha t$

Los pesos varían en la 5ª época y también en la 6ª

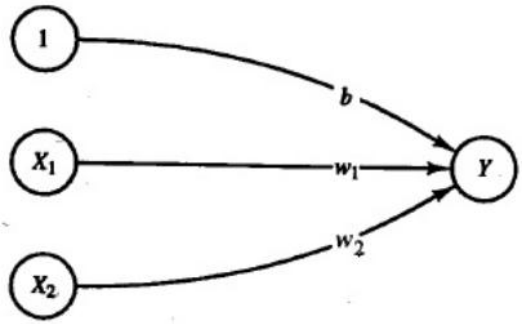
5ª época

x_1	x_2	b	y_in	y	t	Δw_1	Δw_2	Δb	w_1	w_2	b
1	1	1	-2	-1	1	1	1	1	1	2	-2
1	0	1	-1	-1	-1	0	0	0	1	2	-2
0	1	1	0	0	-1	0	-1	-1	1	1	-3
0	0	1	-3	-1	-1	0	0	0	1	1	-3

6ª época

x_1	x_2	b	y_in	y	t	Δw_1	Δw_2	Δb	w_1	w_2	b
1	1	1	-1	-1	1	1	1	1	2	2	-2
1	0	1	0	0	-1	-1	0	-1	1	2	-3
0	1	1	-1	-1	-1	0	0	0	1	2	-3
0	0	1	-3	-1	-1	0	0	0	1	2	-3

Perceptrón para la función AND con entradas binarias y objetivos bipolares



$$w_i(\text{nuevo}) = w_i(\text{anterior}) + \alpha t x_i$$
$$b\ (\text{nuevo}) = b(\text{anterior}) + \alpha t$$

$$y\_in = b + \sum_i x_i w_i$$

$$f(y\_in) = \begin{cases} 1 & \text{si } y\_in > \theta \\ 0 & \text{si } -\theta \le y\_in \le \theta \\ -1 & \text{si } y\_in < -\theta \end{cases}$$

Pesos de la 6ª época:
w_1=1, w_2=2, b=-3
θ= 0.2, α=1
$\Delta w_i = \alpha t x_i$
$\Delta b = \alpha t$

Los pesos varían en la 7ª época y también en la 8ª

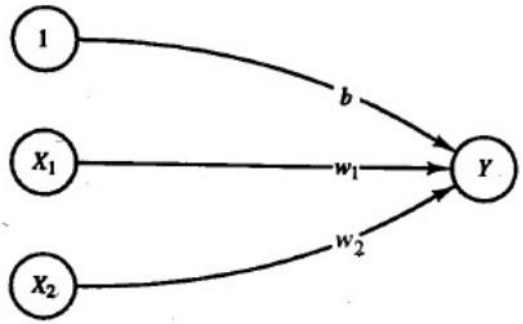
7ª época

x_1	x_2	b	y_in	y	t	Δw_1	Δw_2	Δb	w_1	w_2	b
1	1	1	0	0	1	1	1	1	2	3	-2
1	0	1	0	0	-1	-1	0	-1	1	3	-3
0	1	1	0	0	-1	0	-1	-1	1	2	-4
0	0	1	-4	-1	-1	0	0	0	1	2	-4

8ª época

x_1	x_2	b	y_in	y	t	Δw_1	Δw_2	Δb	w_1	w_2	b
1	1	1	-1	-1	1	1	1	1	2	3	-3
1	0	1	-1	-1	-1	0	0	0	2	3	-3
0	1	1	0	0	-1	0	-1	-1	2	2	-4
0	0	1	-4	-1	-1	0	0	0	2	2	-4

Perceptrón para la función AND con entradas binarias y objetivos bipolares



$$w_i(\text{nuevo}) = w_i(\text{anterior}) + \alpha t x_i$$
$$b\ (\text{nuevo}) = b(\text{anterior}) + \alpha t$$

$$y\_in = b + \sum_i x_i w_i$$

$$f(y\_in) = \begin{cases} 1 & \text{si } y\_in > \theta \\ 0 & \text{si } -\theta \le y\_in \le \theta \\ -1 & \text{si } y\_in < -\theta \end{cases}$$

Pesos de la 8ª época:
w_1=2, w_2=2, b=-4
θ= 0.2, α=1
$\Delta w_i = \alpha t x_i$
$\Delta b = \alpha t$

Los pesos varían en la 9ª pero no en la 10ª, luego termina el aprendizaje.

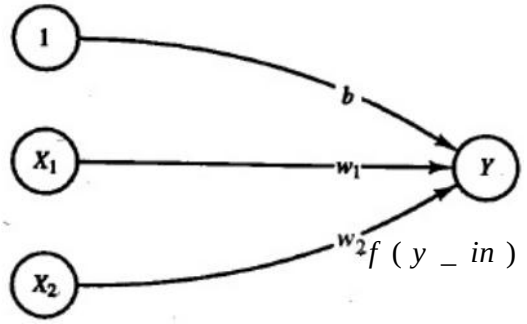
9ª época

x_1	x_2	b	y_in	y	t	Δw_1	Δw_2	Δb	w_1	w_2	b
1	1	1	0	0	1	1	1	1	3	3	-3
1	0	1	0	0	-1	-1	0	-1	2	3	-4
0	1	1	-1	-1	-1	0	0	0	2	3	-4
0	0	1	-4	-1	-1	0	0	0	2	3	-4

10ª época

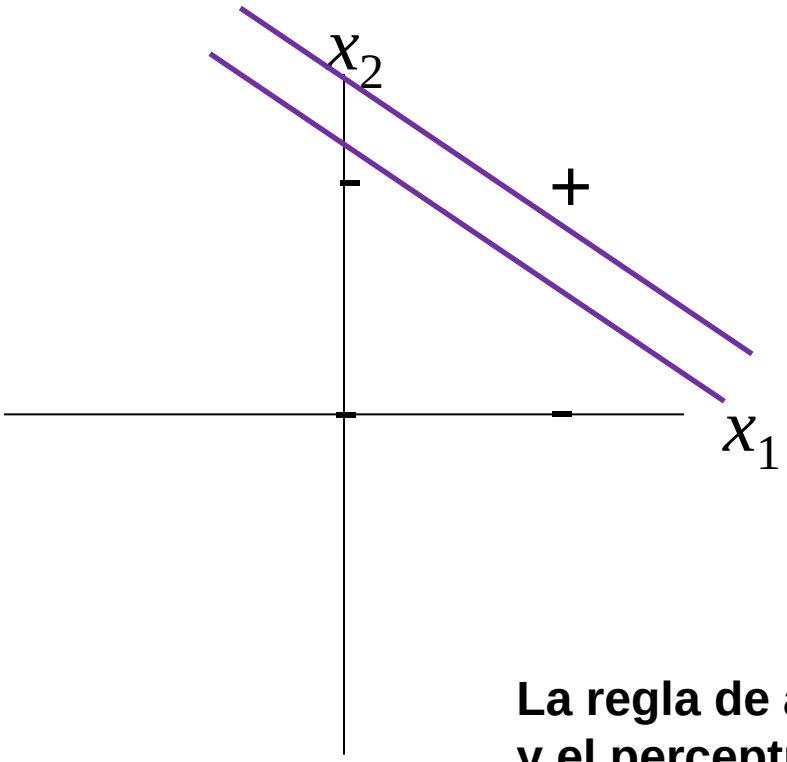
x_1	x_2	b	y_in	y	t	Δw_1	Δw_2	Δb	w_1	w_2	b
1	1	1	1	1	1	0	0	0	2	3	-4
1	0	1	-2	-1	-1	0	0	0	2	3	-4
0	1	1	-1	-1	-1	0	0	0	2	3	-4
0	0	1	-4	-1	-1	0	0	0	2	3	-4

Perceptrón para la función AND con entradas binarias y objetivos bipolares



$$y\_in = b + \sum_i x_i w_i$$

$$f(y\_in) = \begin{cases} 1 & \text{si } y\_in > 0.2 \\ 0 & \text{si } -0.2 \le y\_in \le 0.2 \\ -1 & \text{si } y\_in < -0.2 \end{cases}$$



Después del aprendizaje:

w_1=2; w_2=3; b = -4

La respuesta positiva de la red es para los puntos que cumplen:

$$2x_1+3x_2 -4 > 0.2$$

La línea de decisión es: $x_2 = -\frac{2}{3}x_1 + \frac{7}{5}$

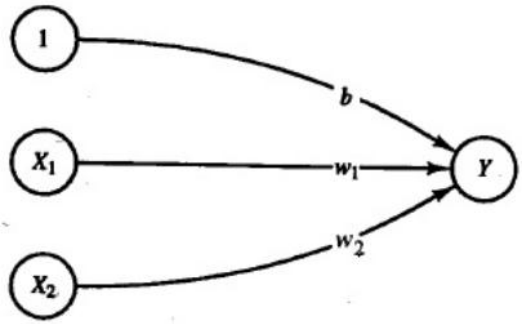
La respuesta negativa de la red es para los puntos que cumplen:

$$2x_1+3x_2 -4 < -0.2$$

La línea de decisión es: $x_2 = -\frac{2}{3}x_1 + \frac{19}{15}$

La regla de aprendizaje ha encontrado los pesos y el perceptrón resuelve el problema.

Perceptrón para la función AND con entradas y objetivos bipolares



$$y\_in = b + \sum_i x_i w_i$$

$$f(y\_in) = \begin{cases} 1 & \text{si } y\_in > \theta \\ 0 & \text{si } -\theta \le y\_in \le \theta \\ -1 & \text{si } y\_in < -\theta \end{cases}$$

w_i(nuevo) = w_i(anterior) + $\alpha t x_i$

b (nuevo) = b(anterior) + αt

Pesos iniciales y sesgo = *0*

θ= 0.2, α=1

$\Delta w_i = \alpha t x_i$

$\Delta b = \alpha t$

1ª época

x_1	x_2	b	y_in	y	t	Δw_1	Δw_2	Δb	w_1	w_2	b
1	1	1	0	0	1	1	1	1	1	1	1
1	-1	1	1	1	-1	-1	1	-1	0	2	0
-1	1	1	2	1	-1	1	-1	-1	1	1	-1
-1	-1	1	-3	-1	-1	0	0	0	1	1	-1

Los pesos han variado en la primera época, luego continuamos con otra.

2ª época

x_1	x_2	b	y_in	y	t	Δw_1	Δw_2	Δb	w_1	w_2	b
1	1	1	1	1	1	0	0	0	1	1	-1
1	-1	1	-1	-1	-1	0	0	0	1	1	-1
-1	1	1	-1	-1	-1	0	0	0	1	1	-1
-1	-1	1	-3	-1	-1	0	0	0	1	1	-1

No varían en la 2ª.

RECORDANDO: Algoritmo de aprendizaje del perceptrón

Paso 0: Inicializar todos los pesos y sesgos (por simplicidad a cero)
Establecer la tasa de aprendizaje $\alpha\ (0 < \alpha \leq 1)$

Paso 1: Mientras que la condición de parada sea falsa, ejecutar pasos 2-6

Paso 2: Para cada par de entrenamiento ($\boldsymbol{s}$:t), ejecutar los pasos 3-5:

Paso 3: Establecer las activaciones a las neuronas de entrada

$$x_i = s_i \quad (i=1 \ldots n)$$

Paso 4: Calcular la respuesta de la neurona de salida:

$$y\_in = b + \sum_i x_i w_i$$

$$f(y\_in) = \begin{cases} 1 & \text{si } y\_in > \theta \\ 0 & \text{si } -\theta \leq y\_in \leq \theta \\ -1 & \text{si } y\_in < -\theta \end{cases}$$

Paso 5: Ajustar los pesos y el sesgo si ha ocurrido un error para este patrón:

Si $y \neq t$

$$w_i(\text{nuevo}) = w_i(\text{anterior}) + \alpha t x_i$$

$$b(\text{nuevo}) = b(\text{anterior}) + \alpha t$$

Si no

$$w_i(\text{nuevo}) = w_i(\text{anterior})$$

$$b(\text{nuevo}) = b(\text{anterior})$$

Paso 6: Comprobar la condición de parada: si no han cambiado los pesos en el paso 2: parar; en caso contrario, continuar.

Uso del Perceptrón (fase de explotación)

Paso 0: Aplicar la regla de aprendizaje para establecer el valor de los pesos de las conexiones.

Paso 1: Para cada vector de entrada $\mathbf{x}$ a clasificar, ejecutar pasos 2-4

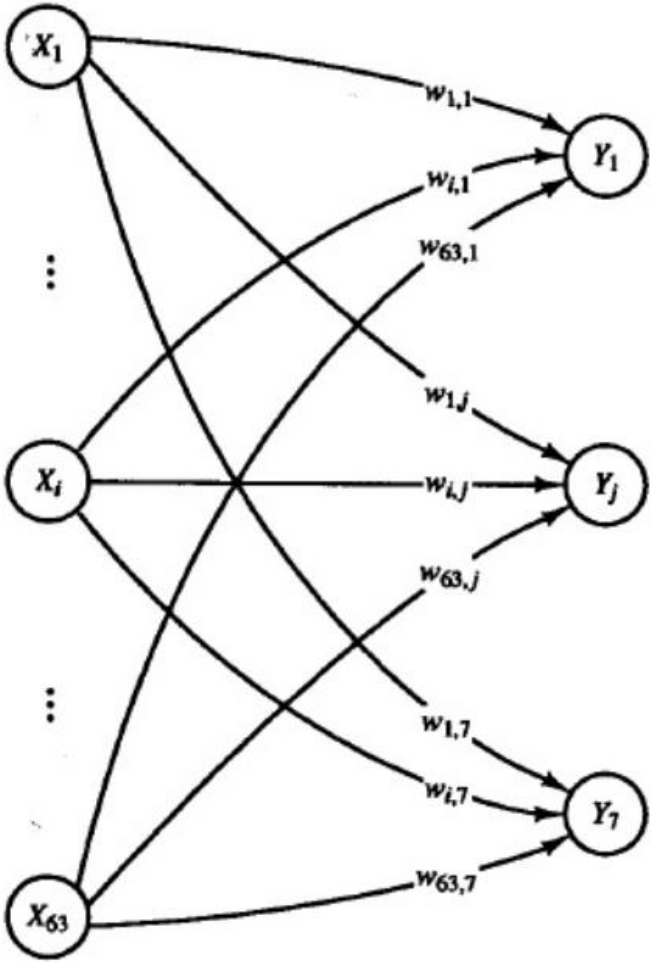
Paso 2: Establecer las activaciones a las neuronas de entrada $x_i = s_i \quad (i=1 \ldots n)$

Paso 3: Calcular la respuesta de la neurona de salida:

$$y\_in = b + \sum_i x_i w_i$$

$$f(y\_in) = \begin{cases} 1 & \text{si } y\_in > \theta \\ 0 & \text{si } -\theta \leq y\_in \leq \theta \\ -1 & \text{si } y\_in < -\theta \end{cases}$$

Perceptrón con varias neuronas de salida



- Las clases de salida pueden ser más de una.
- El perceptrón no cambia de funcionamiento porque haya más de una clase, la regla de aprendizaje se mantiene (los pesos de una neurona de salida no tienen interacción con los de otra neurona de salida).

Algoritmo de aprendizaje del perceptrón: multiples salidas

Paso 0: Inicializar todos los pesos y sesgos (cero o valores pequeños aleatorios)
Establecer la tasa de aprendizaje $\alpha\ (0 < \alpha \leq 1)$

Paso 1: Mientras que la condición de parada sea falsa, ejecutar pasos 2-6

Paso 2: Para cada par de entrenamiento (***s***:*t*), ejecutar los pasos 3-5:

Paso 3: Establecer las activaciones a las neuronas de entrada

$$x_i = s_i \quad (i=1\ldots n)$$

Paso 4: Calcular la respuesta de cada neurona de salida ($j=1\ldots m$):

$$y\_in_j = b_j + \sum_i x_i w_{ij}$$

$$y_j = \begin{cases} 1 & \text{si } y\_in_j > \theta \\ 0 & \text{si } -\theta \leq y\_in_j \leq \theta \\ -1 & \text{si } y\_in_j < -\theta \end{cases}$$

Paso 5: Ajustar los pesos y el sesgo si ha ocurrido un error para este patrón:

Si $y_j \neq t_j$

$$w_{ij}(\text{nuevo}) = w_{ij}(\text{anterior}) + \alpha t_j x_i$$

$$b_j(\text{nuevo}) = b_j(\text{anterior}) + \alpha t_j$$

Si no

$$w_{ij}(\text{nuevo}) = w_{ij}(\text{anterior})$$

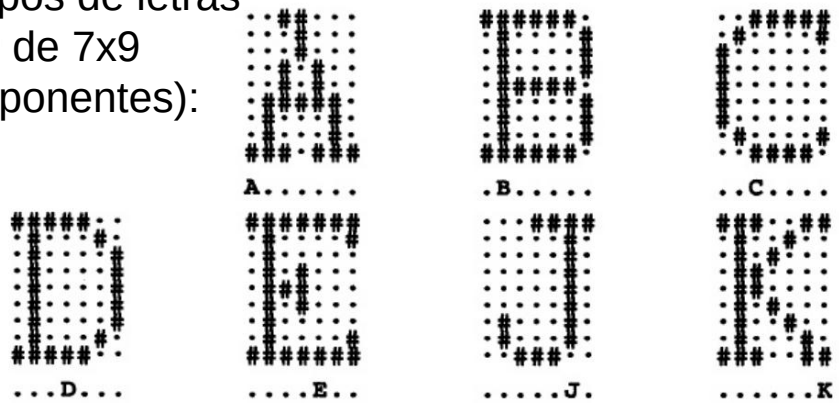
$$b_j(\text{nuevo}) = b_j(\text{anterior})$$

Paso 6: Comprobar la condición de parada: si no han cambiado los pesos en el paso 2: parar; en caso contrario, continuar.

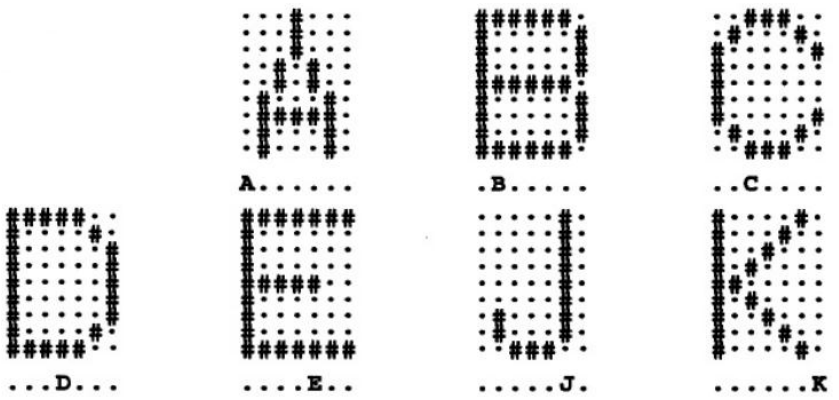
Ejemplo: patrones de entrenamiento

Caracteres de tres tipos de letras distintos en matrices de 7x9 (vectores de 63 componentes):

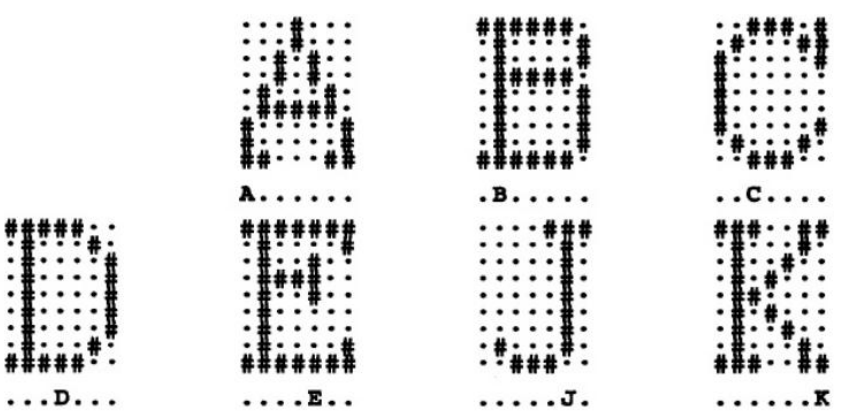
Tipo 1



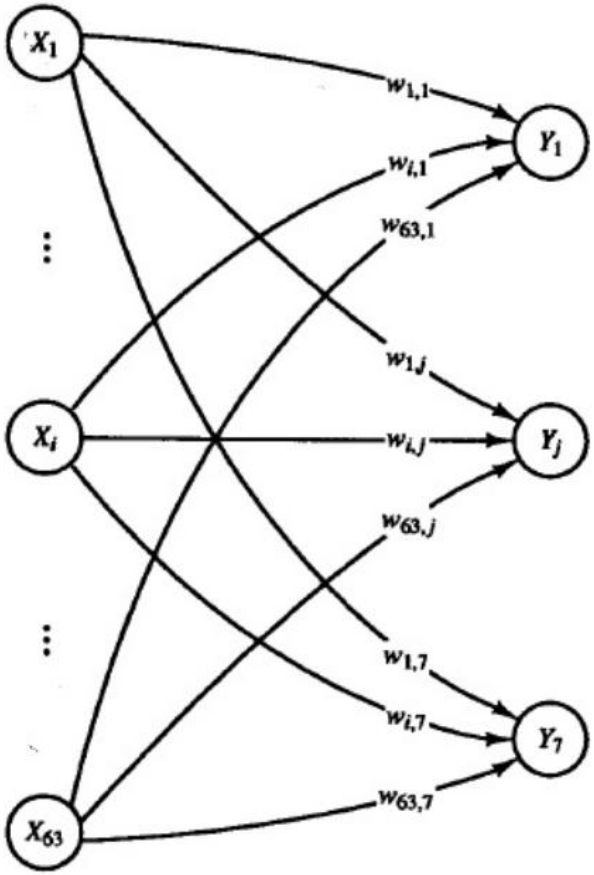
Tipo 2



Tipo 3



7 neuronas de salida para las 7 clases de letras:



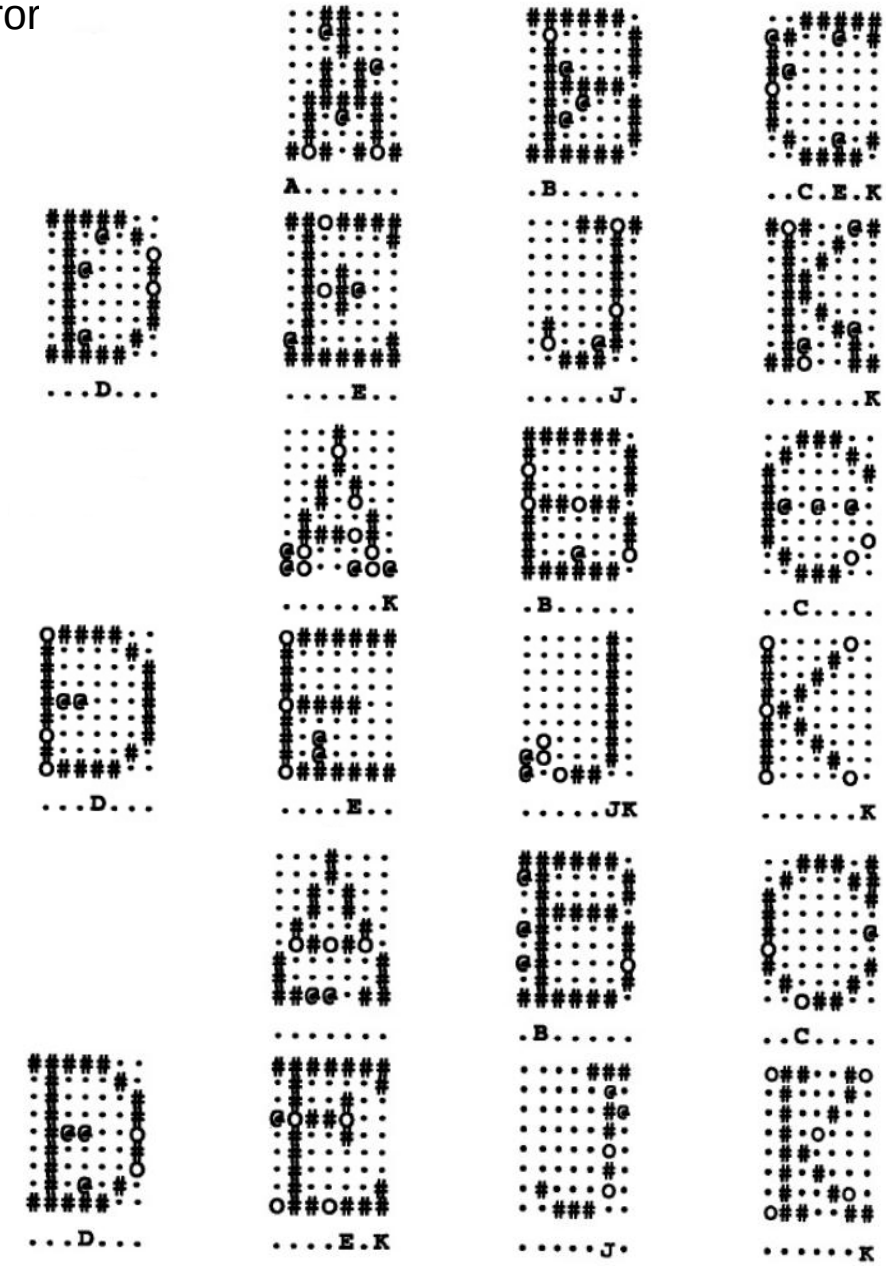
Ejemplo: fase de explotación

Caracteres con error

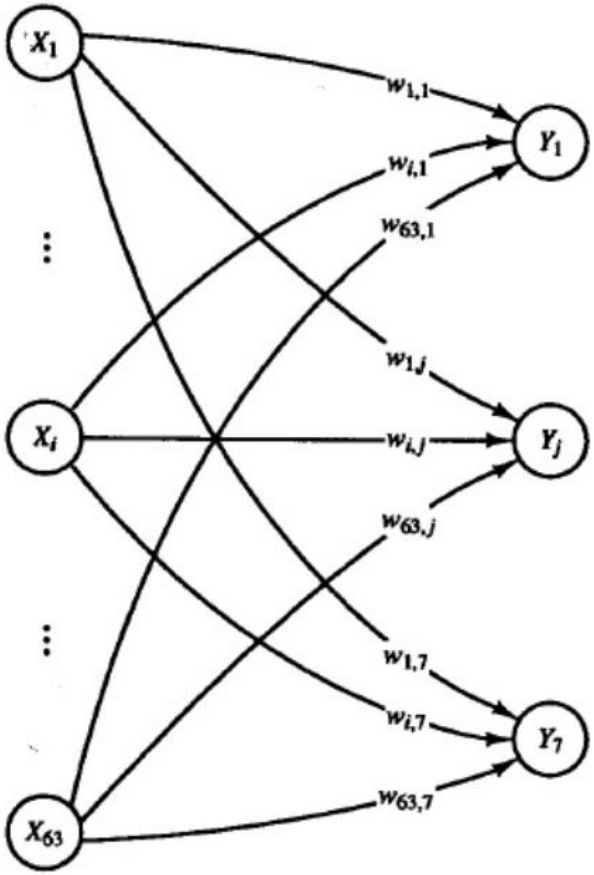
Tipo 1

Tipo 2

Tipo 3



7 neuronas de salida para las 7 clases de letras:



Teorema de convergencia de la regla de aprendizaje del perceptrón

Recordamos la regla de aprendizaje del perceptrón:

Dado un conjunto de P vectores de entrada de entrenamiento $\boldsymbol{x}(p)$, p =1… P,
cada uno con un valor objetivo asociado $t(p)$, p =1… P,
que es ó +1 ó -1, y una función de transferencia $y=f(y\_in)$, con

$$f(y\_in) = \begin{cases} 1 & \text{si } y\_in > \theta \\ 0 & \text{si } -\theta \le y\_in \le \theta \\ -1 & \text{si } y\_in < -\theta \end{cases}$$

los pesos se actualizan de la siguiente forma:

Si $y \neq t$ $\mathbf{w}(nuevo) = \mathbf{w}(antiguo) + t\mathbf{x}$,
Si no, no hay cambio de pesos

Teorema: Si hay un vector de pesos $\mathbf{w}^*$ tal que $f(\mathbf{x}(p)\cdot\mathbf{w}^*)=t(p)$ para todos los p, entonces para cualquier vector de inicio $\mathbf{w}$, la regla de aprendizaje del perceptrón convergerá en un número finito de pasos a un vector de pesos (no necesariamente único y no necesariamente $\mathbf{w}^*$) que proporciona la respuesta correcta para todos los vectores de entrenamiento.